

# Data Structures I:

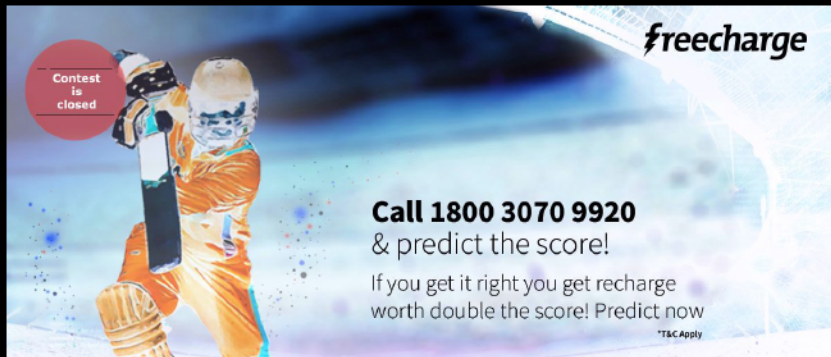
## N-ary trees



Disclaimer: Keep alcohol out of the hands of minors.

- 45 ml of Pisco
- 30 ml of lime juice
- 20 ml of simple syrup
- 1 egg white



A promotional banner for Freecharge featuring a cricket player in an orange uniform swinging a bat. The background is a blurred cricket field. The Freecharge logo is in the top right corner. A red circular badge on the left says 'Contest is closed'. Text on the right encourages calling 1800 3070 9920 to predict the score, with a reward of double the score. A small asterisk and 'T&C Apply' are at the bottom right.

**Contest is closed**

**freecharge**

**Call 1800 3070 9920**  
& predict the score!

If you get it right you get recharge  
worth double the score! Predict now

\*T&C Apply

[https://en.wikipedia.org/wiki/WASP\\_%28cricket\\_calculation\\_tool%29](https://en.wikipedia.org/wiki/WASP_%28cricket_calculation_tool%29)

A binary tree is a tree data structure in which each node has at most two children, which are referred to as the **left child** and the **right child**.

1 Applications

2 Implementation

3 Some Algorithms

- Number of elements

- Maximum Height

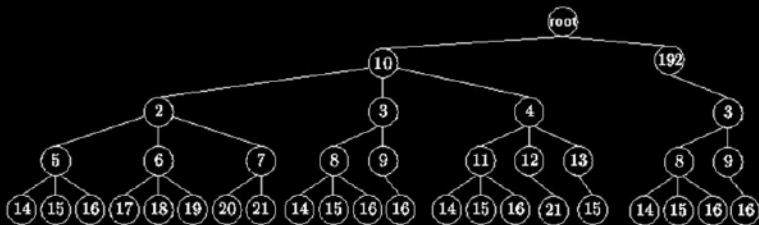
- Tree traversal

- <https://www.youtube.com/watch?v=gm8DUJJhmY4>

- <https://www.youtube.com/watch?v=9RH06jU--GU>



An **N-ary tree** is a rooted tree in which each node has at most  $n$  children.



```
import java.util.ArrayList;
public class Node {
    public final ArrayList<Node> children;
    public final int data;
    public Node(int d){
        data = d;
        children = new ArrayList<Node>();
    }
}
```

```
public class NaryTree {  
    Node root;  
    public NaryTree() {  
        root = null;  
    }  
}
```



```
public class BinaryTree {  
    private int elementsAUX(Node node) {  
        if (node == null)  
            return 0;  
        else  
            return elementsAUX(node.left)+  
                    elementsAUX(node.right)+1;  
    }  
    public int elements() {  
        return elementsAUX(root);  
    }  
}
```

```
public class TernaryTree {  
    private int elementsAUX(Node node) {  
        if (node == null)  
            return 0;  
        else  
            return elementsAUX(node.left)+  
                    elementsAUX(node.center)+  
                    elementsAUX(node.right)+1;  
    }  
    public int elements() {  
        return elementsAUX(root);  
    }  
}
```

```
public class NaryTree {  
    private int elementsAUX(Node node) {  
        int cont = 0;  
        if (node != null)  
            for (int i = 0;  
                i < node.children.size(); i++) {  
                cont++;  
                cont += elementsAUX(  
                    node.children.get(i));  
            }  
        return cont;    }  
    public int elements() {  
        return elementsAUX(root);  
    } }  
}
```

```
public class NaryTree {  
    private int maxHAUX(Node node) {  
        int max = 0;  
        if (node != null)  
            for (int i=0;  
                i<node.children.size(); i++)  
                max = Math.max(  
                    maxHAUX(node.children.get(i))-  
                    max);  
        return max;  
    }  
    public int maxHeight() {  
        return maxHAUX(root);  
    }  
}
```

```
public class NaryTree {  
    private boolean dfsAUX(Node node, int x){  
        boolean resp = false;  
        if (node != null) for (int i = 0;  
            i < node.children.size(); i++) {  
            resp = node.data == x ||  
                dfsAUX(node.children.get(i), x);  
            if (resp) return true;  
        }  
        return resp;  
    }  
    public boolean dfs(int x) {  
        return dfsAUX(root, x);  
    }  
}
```

```
public class NaryTree {  
    private void printAUX(Node node) {  
        if (node != null) {  
            System.out.println(node.data);  
            for (int i = 0;  
                i < node.children.size(); i++)  
                printAUX(node.children.get(i));  
        }  
    }  
    public void print() {  
        printAUX(root);  
    }  
}
```

```
public class NaryTree {  
    private void printByLevels() {  
        LinkedList<Node> queue = new LinkedList();  
        if (root != null)  
            queue.add(root);  
        while (queue.size() != 0) {  
            Node node = queue.pop();  
            System.out.println(node.data);  
            for (int i = 0;  
                i < node.children.size(); i++)  
                if (node.children.get(i) != null)  
                    queue.add(node.children.get(i));  
        }  
    }  
}
```

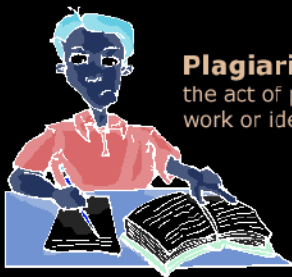
One application is path finding!

[http://kevanahlquist.com/osm\\_pathfinding/](http://kevanahlquist.com/osm_pathfinding/)

Path finding is used in Google Maps and Videogames.



- Please learn how to reference images, trademarks, videos and fragments of code.
- Avoid plagiarism



### **Plagiarism:**

the act of presenting another's work or ideas as your own.

Figure: Figure about plagiarism, University of Malta [Uni09]



University of Malta.

Plagiarism — The act of presenting another's work or ideas as your own, 2009.

[Online; accessed 29-November-2013].

- N-ary trees
  - [https://en.wikipedia.org/wiki/K-ary\\_tree](https://en.wikipedia.org/wiki/K-ary_tree)

